Amendments to the Specification:

Please amend lines 18 and 19 on page 3 of the specification as follows:

~~The State of the software object may correspond to the current values held in the stack, heap, global variables, registers, program counter and the like.~~

When a software object is executed on a computer system, the computer system develops an execution state for the object. In "Modularization and Hierarchy in a Family of Operating Systems", A. Nico Habermann, Lawrence Flon, Lee W. Cooprider, *Commun. ACM 19(5):* 266-272 (1976) ("Habermann"), a software object is called a module, and Habermann teaches that a module is instantiated whenever it is executed by a computer system. The static representation, or text, of a software object is determined at the time it is created, typically by a compiler or assembler. Habermann teaches that an operating system instantiates a module, in part, by allocating some memory locations in the computer system. These dynamically-allocated memory locations are used to store the values in the execution state relating to this particular instantiation. The execution state of a software object is modified by the computer system, when the computer system performs the operations specified by the instructions contained in the software object. It is common in the prior art to distinguish the code section(s) of the text from the data section(s) of the text. Code sections of a software object contain instructions for the computer system, and data sections contain variables referenced by these instructions. The data sections in the text may hold initial values for variables.

In "Heterogeneous process migration by recompilation", M.M. Theimer, B. Hayes, *Proc. 11th Int'l Conf. on Distributed Computing Systems*, 1991, pp. 18-24. DOI: 10.1109/ICDCS.1991 ("Theimer") and "The Use of Program Profiling for Software Maintenance with Applications to the Year 2000 Problem", Thomas W. Reps, Thomas Ball, Manuvir Das, James R. Larus in *Proceedings of the 6th European SOFTWARE ENGINEERING Conference (ESEC/SIGSOFT FSE,* Zurich, 22-25 September 1997*)*, Lecture Notes in Computer Science Vol. 1301, Springer, 1997, pp. 432-449 ("Reps"), an execution state is subdivided into a control state and a data state. The term "process" in this art corresponds to an "instantiated module" in the art of Habermann. Theimer teaches how to migrate a process by transferring its execution state to another computer by "… building a machine-independent migration program that specifies the current code and data state of the process to be migrated. … There are typically three kinds of data space in a program: global data, heap data, and procedure local data." Reps teaches that an execution state can be represented as a combination of code state and data state of the form "(*pt*, *sigma*), where *sigma* is a store value and *pt* is not an arbitrary program point, but one occurring at the beginning of a path *p* that the profiler is prepared to tabulate." A path is a sequence of instructions that were executed from the text. A path may be represented as a series of instruction addresses, i.e. as a series of references to individual instructions in the text. A path may also be represented, more compactly, as "a sequence of edges in the program's control-flow graph" (Reps).

An execution trace is any sequence of execution states or any sequence of partial execution states. A program path *p* in the art of Reps 1997 is an example of a trace. Such a path is sometimes called an "address trace" because it will reveal, to a program analyst, the series of starting addresses of the basic blocks in the program's code that were executed during previous execution states of the program.